

**Aclaraciones:** Cada ejercicio se aprueba **por separado**. Cada hoja debe estar numerada y tener el número de orden y L.U. El parcial dura 4 horas.

- 1) "Porque yo soy tu turrítio  
Y yo sé que quieres ser mi turrítio  
Eres justito lo que necesito  
Yo no andaba buscando una angelita y tú eres una diablita  
Que se está portando mal, serás castigada..."

En el contexto de un proyecto de vinculación entre la academia y la industria (de la música en este caso) somos responsables de idear el nuevo hit de la música urbana. Para esto nos van a dar una lista  $A_1, \dots, A_n$  de palabras que vamos a usar en ese orden. Nuestro trabajo es decidir cómo se particiona la lista de palabras en versos (un verso es un intervalo de palabras consecutivas). En el género evidentemente no tiene mucha importancia lo que las palabras dicen, así que no es necesario tener en cuenta dónde empiezan/terminan las frases/oraciones para separar. Sí es (muy) importante que rime y limitar la cantidad de palabras por verso. Hay dos restricciones:

- Vamos a usar rima parcada. Es decir, la última palabra del primer verso debe rimar con la última del segundo, la del tercer verso con la del cuarto, y así. La cantidad de versos debe ser par.
- Nuestro cantante tiene talento  $c$ : es capaz de cantar hasta  $c$  palabras por verso. Por ende, la cantidad de palabras por verso deberá ser mayor a 0 y a lo sumo  $c$ .<sup>2</sup>

Para hacer que rimen dos palabras que en principio no riman podemos utilizar un "recurso" que consiste en agregarle la terminación "-ito"/"-ita" a una o ambas palabras. Sin embargo, no podemos abusar de esto, porque le produce *cringe* al público. Definimos el *cringe* de la separación en versos como la cantidad de veces que se utiliza el "recurso".

Queremos calcular el mínimo *cringe* que puede lograr una canción generada a partir de la lista de palabras  $A_1, \dots, A_n$  teniendo en cuenta que la va a cantar un cantante de talento  $c$ .

Pueden asumir que tienen programada una función *se\_puede\_cantar\_rima* que dados dos índices  $i, j$  tales que  $1 \leq i < j \leq n$  y el valor  $c$  decide si el intervalo cerrado  $A_i, \dots, A_j$  se puede separar en dos versos válidos que rimen sin modificar palabras. La complejidad de esta función es lineal en el tamaño del intervalo.

- Definir en forma recursiva la función  $f_c: \mathbb{N} \rightarrow \mathbb{N}$  tal que  $f_c(i)$  devuelve el mínimo valor de *cringe* alcanzable para el sufijo de palabras  $A_i, \dots, A_n$  con un cantante de talento  $c$ . Indicar qué llamado(s) hay que hacer a esta función para resolver el problema. **Importante:** acompañen a la definición recursiva con una explicación en castellano.
- Demostrar que  $f_c$  tiene la propiedad de superposición de subproblemas.
- Definir un algoritmo *top-down* para calcular  $f_c(i)$  indicando claramente las estructuras de datos utilizadas y la complejidad resultante. Escribir el (pseudo-)código del algoritmo.

**Criterio de aprobación:** La función descrita en el inciso a) debe ser correcta y estar adecuadamente justificada, y la complejidad temporal del algoritmo resultante para computar  $f_c(i)$  en el inciso c) debe ser  $O(n * c^2)$  (y también estar justificada).

- 2) Dado un grafo conexo  $G = (V, E)$ , decimos que  $v \in V$  es un *punto de articulación* de  $G$  si el grafo que se obtiene al quitar  $v$  de  $G$  es desconexo. Más formalmente,  $v$  es punto de articulación de  $G$  si el subgrafo de  $G$  inducido por los nodos  $V \setminus \{v\}$  es desconexo.

Sea  $T_r = (V, E_T)$  un árbol DFS de  $G$  enraizado en  $r$ .

<sup>1</sup>Incluyendo esta hoja.

<sup>2</sup>El valor  $c$  es parte del input, no una constante.



- Probar que  $r$  es un punto de articulación de  $G$  si y solamente si  $r$  tiene grado mayor a 1 en  $T_r$ .
- Dar un criterio simple basado en las *back edges* de  $T_r$  para determinar si un nodo  $v \neq r$  es punto de articulación. Justificar.
- Diseñar un algoritmo que dado un grafo  $G = (V, E)$  encuentre todos los puntos de articulación de  $G$  en tiempo  $O(|V| + |E|)$ . Argumentar su correctitud y complejidad temporal.
- Diseñar un algoritmo que dado un grafo  $G = (V, E)$  construya en tiempo  $O(|V| + |E|)$  una estructura de datos capaz de responder, dados dos nodos  $v$  y  $w$ , si existe un camino entre  $v$  y  $w$  que no pase por ningún punto de articulación en  $O(1)$ . Explicar cómo se responde cada consulta.

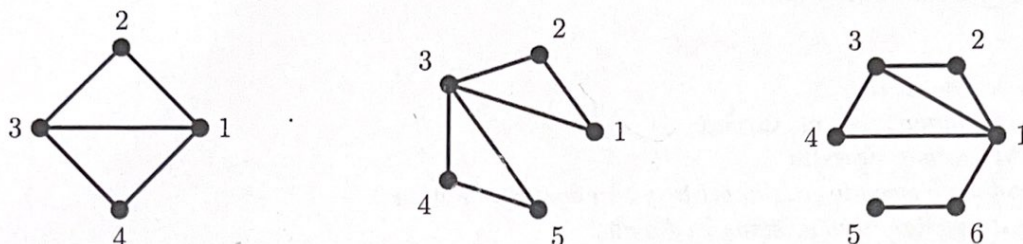


Figura 1: El grafo de la izquierda no tiene puntos de articulación. Mientras tanto, el grafo del centro tiene un único punto de articulación: el nodo 3. Finalmente, el grafo de la derecha tiene dos puntos de articulación: los nodos 1 y 6.

**Criterio de aprobación:** Por lo menos 3 de los incisos tienen que estar bien resueltos.

- En el país de *Nlogonia* hay  $n$  ciudades. El gobernador quiere realizar una red eléctrica de tal forma que se pueda abastecer con energía a todas las ciudades, y para lograr esto puede construir *tendidos* eléctricos uniendo las distintas ciudades (cada tendido une a un único par de ciudades). El gobernador tiene un orden de preferencia sobre los *tendidos*, y por lo tanto sus costos son todos **distintos**. Además, estos costos son siempre potencia de 2. El gobernador sabe calcular el costo de armar la red de **mínimo** costo pero, para no estar ajustado de presupuesto, quiere el costo de la red que sea **mayor al mínimo** pero menor a cualquier otro. Decimos que busca el **segundo costo mínimo** entre todos los árboles generadores.

Sea  $G$  el grafo pesado completo que representa al país de *Nlogonia*, donde cada nodo es una ciudad y cada eje  $vw$  representa el *tendido* entre  $v$  y  $w$  con su costo.

- Probar que un árbol de segundo costo mínimo de  $G$  se obtiene tomando un AGM  $T$  de  $G$  e intercambiando el eje de menor peso de  $G$  que no está en  $T$  con alguno de  $T$ .<sup>3</sup>
- Empleando la idea del inciso a), dar un algoritmo que devuelva el **segundo costo mínimo** de  $G$  y justificar su correctitud. El algoritmo debe tener complejidad  $O(n^2)$ .

Luego de presentar el proyecto el gobernador tuvo algunos rechazos con distintos sectores. En particular le hicieron notar que los costos que asignó toman valores tan grandes que no son soportados por las computadoras que tienen disponibles. Por lo tanto decidió que en vez de ser potencias de dos, los costos serán multiplos de dos y seguirán siendo todos **distintos**. No está seguro si le sirve la solución que le presentamos anteriormente y nos pide ayuda para evaluarlo.

- Decidir si se puede utilizar el algoritmo del inciso b). Justificar en caso positivo o dar un contraejemplo en caso negativo.

El gobernador está convencido de que el problema ya estaba resuelto. Propone que con alguna corrección en la asignación de costos podríamos utilizar el algoritmo presentado. Para esto se define la operación de *swap*, que consiste en tomar dos tendidos e intercambiarles el costo.

- Dar un algoritmo de complejidad  $O(n^2)$  que dado  $G$  indique qué aristas *swappear* para que el algoritmo de b) encuentre el segundo costo mínimo. Justificar su correctitud y complejidad.

**Criterio de aprobación:** Por lo menos 3 de los incisos tienen que estar bien resueltos.